REMARKS

This Amendment is responsive to the Office Action dated October 3, 2005. Applicant has amended claims 1, 5, 12, 20, 23, 24, and 25. Claims 1-28 are pending.

Claim Rejection Under 35 U.S.C. § 102

In the Office Action, the Examiner rejected claims 1-7 and 22 under 35 U.S.C. 102(e) as being anticipated by Hankison et al. (USPN 6,799,202), and rejected claims 12-15, 17-21, 23 and 24 under 35 U.S.C. 102(e) as being anticipated by Abjanic (USPN 6,732,175). The Examiner also rejected claims 8-11 under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. in view of Gelman et al. (USPN 6,415,329), and rejected claim 16 as being unpatentable over Abjanic et al. in view of Gelman et al.. Applicant respectfully traverses these rejections to the extent such rejections may be considered applicable to the amended claims.

Claim amendments

Applicant's claim 1 is directed to a load-balancing network acceleration device that comprises both (i) an encryption and decryption engine, and (ii) a load balancing engine. Applicant has amended claim 1 to limit the claim to embodiments in which the decryption engine and the load balancing engine associates provide process encrypted communications and load balance client devices with server devices *without processing the data packets with an application layer of a network stack*. In particular, amended claim 1 requires that the decryption engine and the load balancing engine *bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.*

Applicant has made similar claim amendments to independent claims 12 and 25 For example, amended claim 12 requires the step of, selecting with the acceleration device at least one of the plurality of servers in the enterprise and associating the selected server with a communications session from the one of the clients *without processing the data packets with an application layer of a network stack.*

For purposes of clarity, Applicants refer the Examiner to Figure 2B and pages 5 and 6 that describe conventional SSL acceleration devices. For the convenience of the Examiner, Figure 2B is reproduced below:

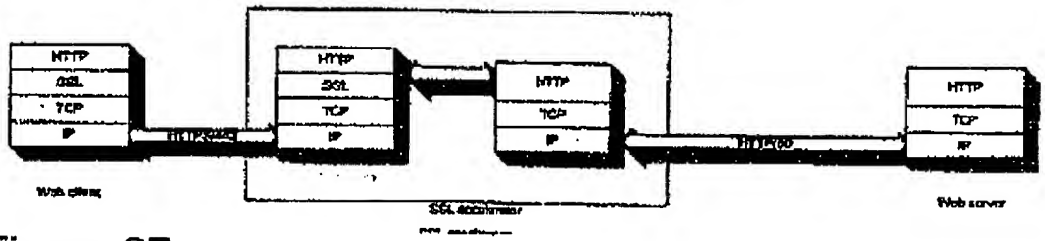

Figure 2B

Applicant draws the Examiner's attention to the SSL accelerator shown above (middle device). As illustrated in FIG. 2B, prior art acceleration systems process HTTP packets through the entire networking protocol stack up to and including the application layer (HTTP layer) in order to provide network acceleration or load balancing. The HTTP packets are reassembled at the application layer in order to extract the HTTP headers or the application data. As explained in Applicant's Background, the HTTP packets travel through the entire TCP stack, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator. This also requires a great deal of random access memory, usually around 8-10kB per TCP session, for retransmission support. This type of architecture also has scalability and fault tolerance problems because all of the TCP and SSL state databases are concentrated on one SSL accelerator device.

In contrast, Applicant's claims as amended are directed to an acceleration device that *bypasses the application layer* of the network stack when providing load balancing and decryption of secure messages. Applicant refers the Examiner to FIG. 3 of the present application and the related description that describes how the claimed embodiments of the present invention differ from the prior art by supporting a "cut through" processing mode in which packets are intercepted and load balanced *without transmitting the packets up the TCP/IP stack to the application layer*.

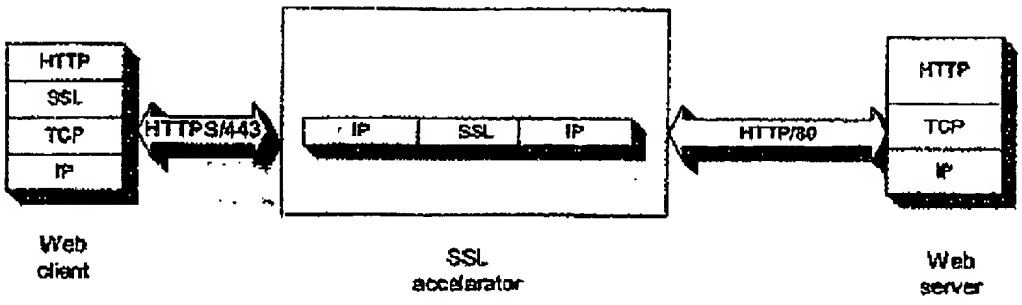

FIGURE 3

Again, Applicant draws the Examiner's attention to the intermediate SSL accelerator above that, as discussed in detail in the present application, does not process the packets at the application layer to reassemble application data. Noticeably, the packets are not processed at the HTTP protocol to extract application data. As discussed above, Applicant has amended the independent claims to direct the claims to an intermediate network acceleration device that *bypasses the application layer of the network stack* performs load balancing of secure communications *without processing the data packets with an application layer of a network stack*.

Abjanic

Abjanic describes a network apparatus located between a network and a plurality of processing nodes or servers. The Abjanic network apparatus includes a content-based message director (e.g., *XML* director) to route or direct messages received from the network to one of the processing nodes *based upon the application data*, including business transaction information.[1] Abjanic makes clear that the described apparatus is an application-layer content-based switching apparatus. Abjanic provides numerous examples of how application layer data (XML data in this case) is extracted using the HTTP protocol (which is an application-layer protocol) in order to make forwarding decisions. Below is one brief example:

> *The application data is provided after the HTTP header, and in this example is provided as XML data. ... [T]he present invention is directed to a technique to perform*

[1] Abjanic at Summary.

switching at a network apparatus based upon the application data, such as XML data (which includes business transaction information).[2]

HTTP headers and XML data are only available at the application layer. Abjanic fails to teach or suggest mechanisms by which a load balancing acceleration device can decrypt client requests and associate client devices with server devices (i.e., load balance) *by bypassing the application layer without processing the decrypted data with an application layer of a network stack.* Rather, like the prior art discussed by the Applicant's Background, the Abjanic appliance requires assembly of application data in order to make switching decisions.

For at least these reasons, Abjanic in view of the other references fails to teach or suggest each and every limitation set forth in Applicant's claims.

Hankison et al.

Hankinson describes techniques for implementing a distributed, high capacity, high speed, operating system.[3] According to Hankinson, the operating system may be incorporated into a web server having a plurality of "members." Each "member" has a distinct specialized operating system that is optimized for its function.[4] Thus, taken as a whole, Hankinson describes a web server in which multiple operation systems are used to perform functions, and a function may be implemented by operating systems executing on one or many different servers.

With respect to load balancing, Hankinson describes how the load of performing a networking function may be "inherently" distributed across the different operating systems. Hankinson provides the example of a TCP/IP state machine, where execution of the state machine itself is distributed across the operating systems. That is, the plurality of operating systems cooperate to provide the TCP/IP function. Similarly, with respect to encryption, Hankinson notes that a member (i.e., an operating system) may support SSL. There is no teaching or suggestion that SSL is supported in a manner that is different from the prior art, i.e., where application data be reassembled via the application layer for decryption (see FIG. 2B of Applicant's Background).

---

[2] Abjanic at Col. 6, ll. 1-25
[3] Hankinson at Summary.
[4] Id.

For at least these reasons, the Hankinson federated operating system for web servers does not describe or suggest an intermediate load balancing device that decrypts requests and then load balances those requests across servers without processing the decrypted data of those requests at the application layer. In fact, the Hankinson federated operating system is not load balancing decrypted client requests at all. Rather, the federated operated system distributes the load of performing a task, such as implementing TCP/IP or implementing the function of SSL using the different operating systems and different computing resources. Hankinson does not teach or suggest a manner for load balancing decrypted data from encrypted client requests without processing the decrypted data at the application layer.

For at least these reasons, Hankinson et al. in view of the other references fails to teach or suggest each and every limitation set forth in Applicant's claims.

Baskey

In general, Baskey describes an SSL proxy server 40 that acts as a proxy server for a transaction server 50. In col. 6, ll. 17-35, Baskey makes mention that the SSL proxy server 40 may serve multiple transaction servers 50. However, directly counter to Applicant's claims as amended, Baskey states that the routing function 42 may be provided in the *application layer* of a protocol as an application program which receives information from the SSL of a first protocol stack and retransmits the information to a second SSL connection over a *second protocol stack*. This typically approach for a full proxy requires to full networking stacks. Baskey fails to describe any other mechanism for implementing the SSL proxy.

For at least these reasons, the cited prior art fails to establish a prima facie case for non-patentability of Applicant's claims under 35 U.S.C. 102 or 103(a). Withdrawal of the rejections is requested.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: February 3, 2006

By: _____

SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Kent J. Sieffert
Reg. No.: 41,312